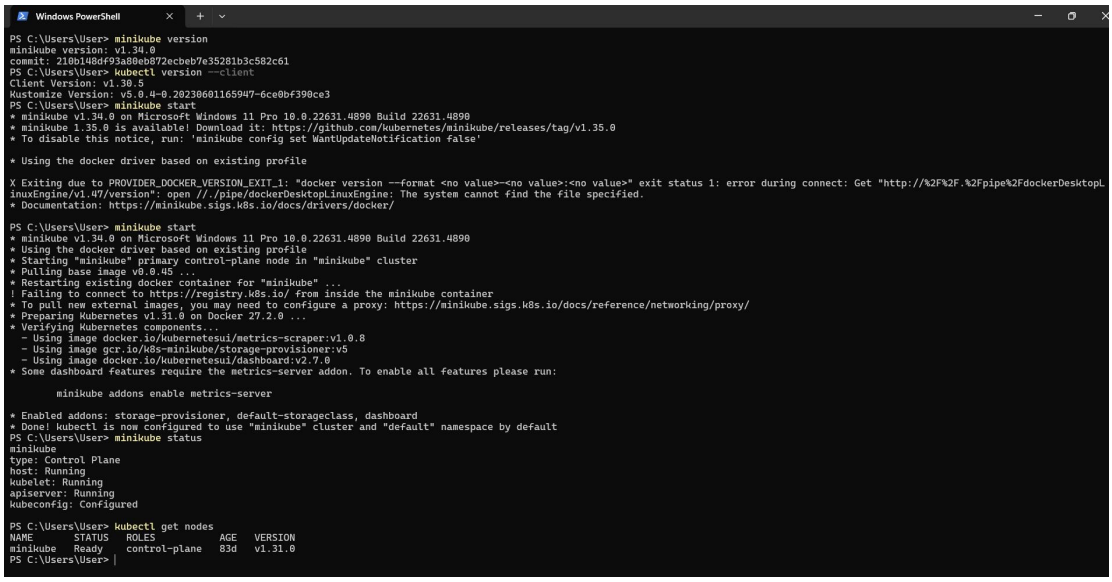
Practical Task 1: Set Up a Local Kubernetes Environment
Requirements:

• Install a local Kubernetes environment using **Minikube** or **Docker Desktop with**
Kubernetes enabled.

• Verify the installation by running **kubectl** commands to check the cluster status.

• Deploy a simple "Hello World" application using a Deployment resource.• Expose the application using a Service of type **NodePort** and verify access from your local
machine.



```
Windows PowerShell
PS C:\Users\User> minikube version
minikube version: v1.34.0
commit: 210b148df93a80eb872ecbeb7e35281b3c582c61
PS C:\Users\User> kubectl version --client
Client Version: v1.30.5
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
PS C:\Users\User> minikube start
* minikube v1.34.0 on Microsoft Windows 11 Pro 10.0.22631.4890 Build 22631.4890
* minikube 1.35.0 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.35.0
* To disable this notice, run: 'minikube config set WantUpdateNotification false'

* Using the docker driver based on existing profile

X Exiting due to PROVIDER_DOCKER_VERSION_EXIT_1: "docker version --format <no value>-<no value>:<no value>" exit status 1: error during connect: Get "http://%2F%2F.%2Fpipe%2FdockerDesktopL
inuxEngine/v1.47/version": open //./pipe/dockerDesktopLinuxEngine: The system cannot find the file specified.
* Documentation: https://minikube.sigs.k8s.io/docs/drivers/docker/

PS C:\Users\User> minikube start
* minikube v1.34.0 on Microsoft Windows 11 Pro 10.0.22631.4890 Build 22631.4890
* Using the docker driver based on existing profile
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Restarting existing docker container for "minikube" ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
* Verifying Kubernetes components...
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
* Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

* Enabled addons: storage-provisioner, default-storageclass, dashboard
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\Users\User> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Users\User> kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   83d   v1.31.0
PS C:\Users\User>
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: hello-world-service
spec:
  type: NodePort
  selector:
    app: hello-world
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30010
```

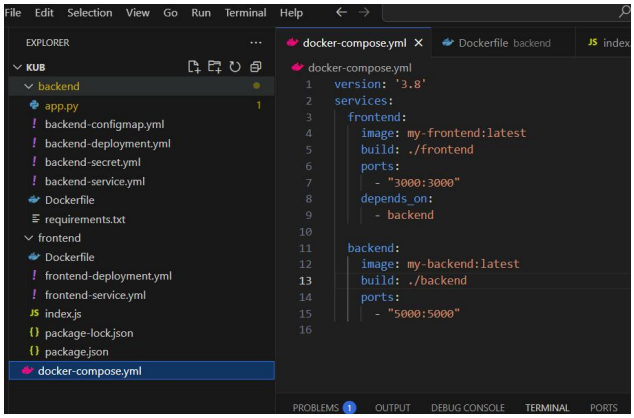
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - name: hello-world
        image: nginxdemos/hello
        ports:
        - containerPort: 80
```

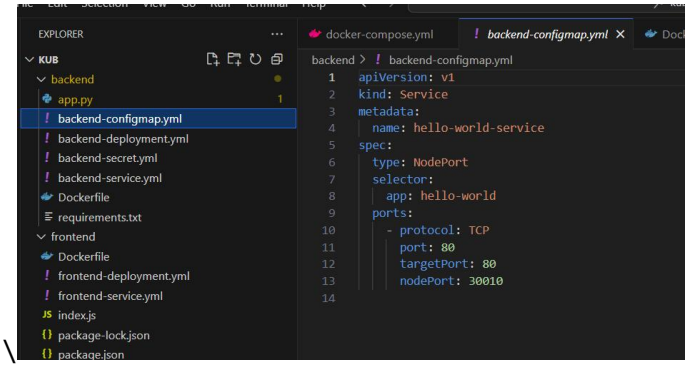
```
PS E:\projects\terraformProj\DevopsService\kub> kubectl apply -f deployment.yml
deployment.apps/hello-world unchanged
PS E:\projects\terraformProj\DevopsService\kub> kubectl get deployments
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
api-gateway             0/1     1            0           76d
api-gateway-deployment  0/1     1            0           76d
backend-deployment      0/1     1            0           80d
frontend-deployment     0/1     1            0           80d
hello-world             1/1     1            1           4m7s
my-nginx-deployment     0/2     2            0           80d
worker                  0/1     1            0           76d
PS E:\projects\terraformProj\DevopsService\kub> kubectl apply -f service.yml
The Service "hello-world-service" is invalid: spec.ports[0].nodePort: Invalid value: 30007: provided port is already allocated
PS E:\projects\terraformProj\DevopsService\kub> kubectl get services -o wide
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE   SELECTOR
api-gateway-service NodePort    10.110.241.61   <none>        8080:30001/TCP   76d   app=api-gateway
backend-service     ClusterIP   10.104.40.174   <none>        5000/TCP         80d   app=backend
frontend-service    NodePort    10.99.106.47    <none>        80:31340/TCP     80d   app=frontend
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP          83d   <none>
my-nginx-service    NodePort    10.98.223.222   <none>        80:30007/TCP     80d   app=my-nginx
worker-service      ClusterIP   10.102.76.12    <none>        8081/TCP         76d   app=worker
PS E:\projects\terraformProj\DevopsService\kub> kubectl apply -f service.yml
service/hello-world-service created
PS E:\projects\terraformProj\DevopsService\kub> kubectl get services
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
api-gateway-service NodePort    10.110.241.61   <none>        8080:30001/TCP   76d
backend-service     ClusterIP   10.104.40.174   <none>        5000/TCP         80d
frontend-service    NodePort    10.99.106.47    <none>        80:31340/TCP     80d
```

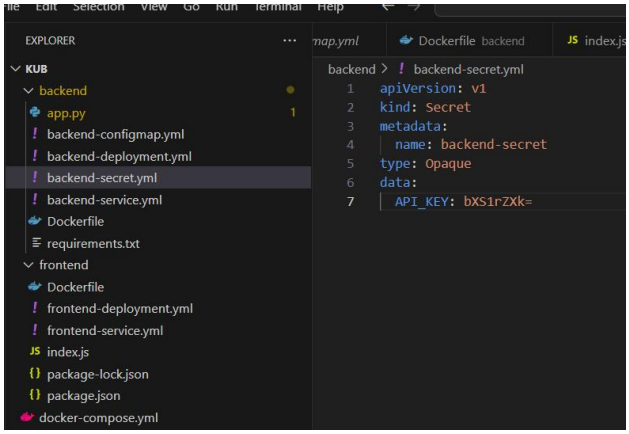
```
PS E:\projects\terraformProj\DevopsService\kub> kubectl apply -f service.yml
service/hello-world-service created
PS E:\projects\terraformProj\DevopsService\kub> kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
api-gateway-service   NodePort    10.110.241.61   <none>        8080:30001/TCP   76d
backend-service       ClusterIP   10.104.40.174   <none>        5000/TCP         80d
frontend-service      NodePort    10.99.106.47    <none>        80:31340/TCP     80d
hello-world-service   NodePort    10.104.184.23   <none>        80:30010/TCP     14s
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          83d
my-nginx-service      NodePort    10.98.223.222   <none>        80:30007/TCP     80d
worker-service        ClusterIP   10.102.76.12    <none>        8081/TCP         76d
PS E:\projects\terraformProj\DevopsService\kub> minikube ip
192.168.49.2
```

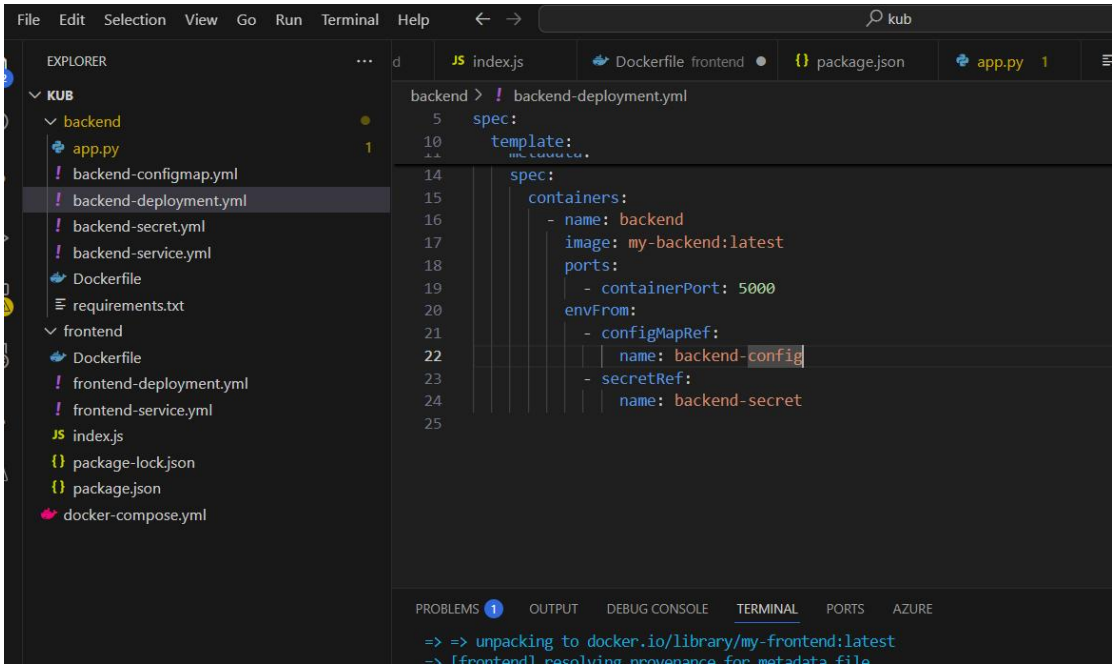
```
PS E:\projects\terraformProj\DevopsService\kub> minikube service hello-world-service --url
http://127.0.0.1:22852
❗  Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

Practical Task 2: Deploy a Multi-Container Application

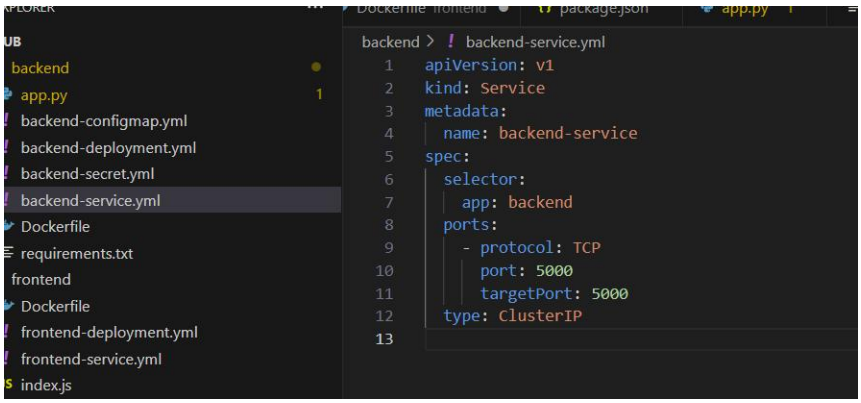



EXPLORER
KUB
backend
app.py
backend-configmap.yml
backend-deployment.yml
backend-secret.yml
backend-service.yml
Dockerfile
requirements.txt
frontend
Dockerfile
frontend-deployment.yml
frontend-service.yml
index.js
package-lock.json
package.json
docker-compose.yml
backend-configmap.yml
backend > backend-configmap.yml
apiVersion: v1
kind: Service
metadata:
name: hello-world-service
spec:
type: NodePort
selector:
app: hello-world
ports:
- protocol: TCP
port: 80
targetPort: 80
nodePort: 30010

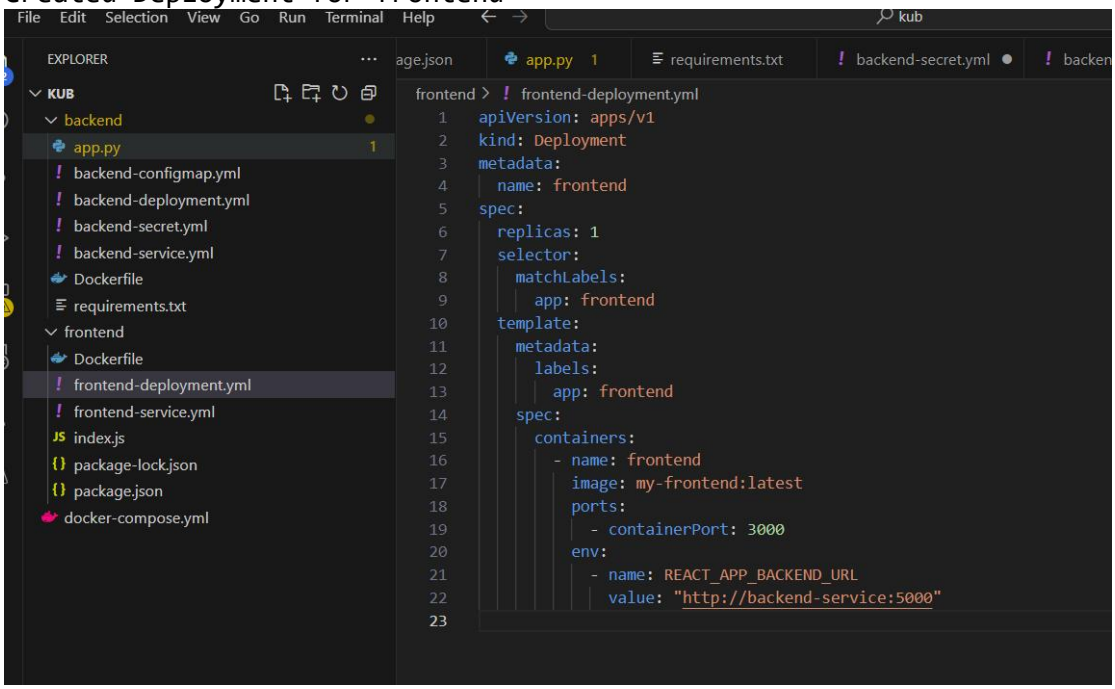

EXPLORER
KUB
backend
app.py
backend-configmap.yml
backend-deployment.yml
backend-secret.yml
backend-service.yml
Dockerfile
requirements.txt
frontend
Dockerfile
frontend-deployment.yml
frontend-service.yml
index.js
package-lock.json
package.json
docker-compose.yml
Dockerfile backend
backend > backend-secret.yml
apiVersion: v1
kind: Secret
metadata:
name: backend-secret
type: Opaque
data:
API_KEY: bXS1rZXk=


File Edit Selection View Go Run Terminal Help
kub
EXPLORER
KUB
backend
app.py
backend-configmap.yml
backend-deployment.yml
backend-secret.yml
backend-service.yml
Dockerfile
requirements.txt
frontend
Dockerfile
frontend-deployment.yml
frontend-service.yml
index.js
package-lock.json
package.json
docker-compose.yml
index.js
Dockerfile frontend
package.json
app.py
backend > backend-deployment.yml
spec:
template:
spec:
containers:
- name: backend
image: my-backend:latest
ports:
- containerPort: 5000
envFrom:
- configMapRef:
name: backend-config
- secretRef:
name: backend-secret
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
=> => unpacking to docker.io/library/my-frontend:latest

Created Service for backend



Created Deployment for frontend



Created for Service frontend

File Edit Selection View Go Run Terminal Help

EXPLORER

KUB

- backend
  - app.py 1
  - backend-configmap.yml
  - backend-deployment.yml
  - backend-secret.yml
  - backend-service.yml
  - Dockerfile
  - requirements.txt
- frontend
  - Dockerfile
  - frontend-deployment.yml
  - frontend-service.yml
  - index.js
  - package-lock.json
  - package.json
- docker-compose.yml

requirements.txt | backend-secret.yml

frontend > frontend-service.yml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: NodePort
```

EXPLORER

KUB

- backend
  - app.py 1
  - backend-configmap.yml
  - backend-deployment.yml
  - backend-secret.yml
  - backend-service.yml
  - Dockerfile
  - requirements.txt
- frontend
  - node_modules
  - Dockerfile
  - frontend-deployment.yml
  - frontend-service.yml
  - index.js
  - package-lock.json
  - package.json
- docker-compose.yml

index.js | Dockerfile frontend | package-lock.json | package.json | app.py 1 | requirements.txt | backend-secret.yml

frontend > index.js > ...

```js
const express = require("express");
const app = express();

app.get("/", (req, res) => {
  res.send("Frontend is running!");
});

app.listen(3000, "0.0.0.0", () => {
  console.log("Frontend is running on port 3000");
});
```

PROBLEMS 1 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | AZURE | powershell

```
 => [frontend 4/5] RUN npm install
 => [frontend 5/5] COPY . .
 => [frontend] exporting to image
 => => exporting layers
 => => exporting manifest sha256:3d4e28ef671a6d63d815224d45928dc07b4c5aaf8e3556aa60a5e831df1fd7fb
 => => exporting config sha256:194f3ece1e2899251a135596d729062e295f35823dd8a526ebd65ede8283e28f
 => => exporting attestation manifest sha256:a2677d8d86e7ad6841a3ac71c0cf9947aa63111e7da39eaa5ba076f2aa663d6d
 => => exporting manifest list sha256:1e7b6a69977515d8415dfad40e410e4054801487ebd27aa2ade53ce4597631f6
 => => naming to docker.io/library/my-frontend:latest
 => => unpacking to docker.io/library/my-frontend:latest
 => [frontend] resolving provenance for metadata file
[+] Running 2/2
 ✓ Container kub-frontend-1  Started
 ✓ Container kub-backend-1  Started
PS E:\projects\terraformProj\DevopsService\kub>
PS E:\projects\terraformProj\DevopsService\kub> docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS         PORTS                    NAMES
2e083ae6e303   my-frontend:latest   "docker-entrypoint.s…"   8 seconds ago    Up 7 seconds   0.0.0.0:3000->3000/tcp   kub-frontend-1
fbf4c84d76f7   my-backend:latest    "python app.py"          18 seconds ago   Up 7 seconds   0.0.0.0:5000->5000/tcp   kub-backend-1
PS E:\projects\terraformProj\DevopsService\kub>
```

OUTLINE

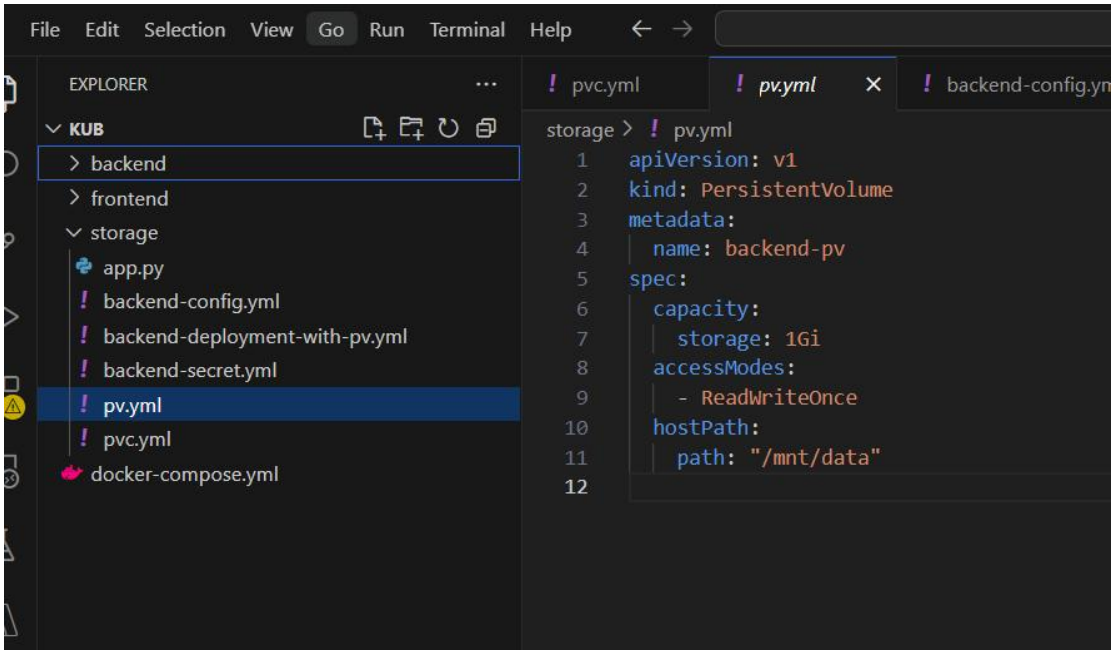
TIMELINE

0 ⚠ 1 | Ln 11, Col 1 | Spaces: 4 | UTF-8 | CRLF | {}

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   AZURE                    powershell - frontend

PS E:\projects\terraformProj\DevopsService\kub> curl http://localhost:5000


StatusCode        : 200
StatusDescription : OK
Content           : Backend is working!
RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 19
                    Content-Type: text/html; charset=utf-8
                    Date: Thu, 06 Mar 2025 12:50:16 GMT
                    Server: Werkzeug/3.1.3 Python/3.9.21

                    Backend is working!
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 19], [Content-Type, text/html; charset=utf-8], [Date, Thu, 06 Mar 2025 12:50:16 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 19
```

Frontend is running!

Practical Task 3: Implement Persistent Storage with Persistent Volumes



EXPLORER

KUB

- backend
- frontend
- storage
  - app.py
  - backend-config.yml
  - backend-deployment-with-pv.yml
  - backend-secret.yml
  - pv.yml
  - pvc.yml
- docker-compose.yml

pvc.yml | pv.yml | backend-config.yml

storage > pvc.yml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: backend-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

EXPLORER

KUB

- backend
  - app.py
  - backend-configmap.yml
  - backend-deployment.yml
  - backend-secret.yml
  - backend-service.yml
  - Dockerfile
  - requirements.txt
- frontend
- storage
  - app.py
  - backend-config.yml
  - backend-deployment-with-pv.yml
  - backend-secret.yml
  - pv.yml
  - pvc.yml
- docker-compose.yml

OUTLINE

pvc.yml | backend-config.yml | backend-secret.yml | app.py | backend-configmap.yml | backend-deployment-with-pv.yml

storage > backend-deployment-with-pv.yml

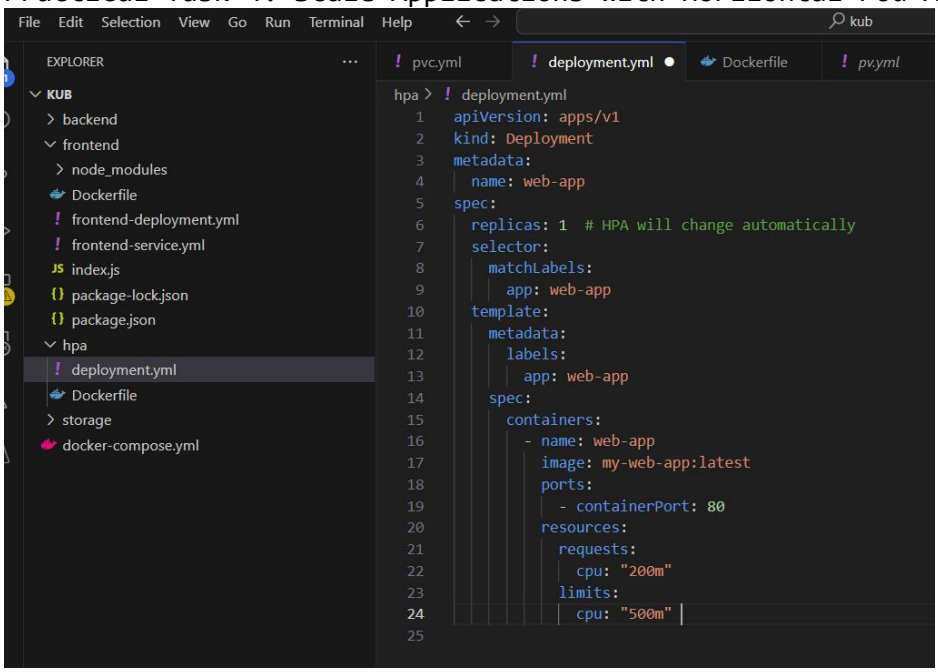
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend
          image: my-backend:latest
          volumeMounts:
            - name: logs-storage
              mountPath: /var/log/app
```

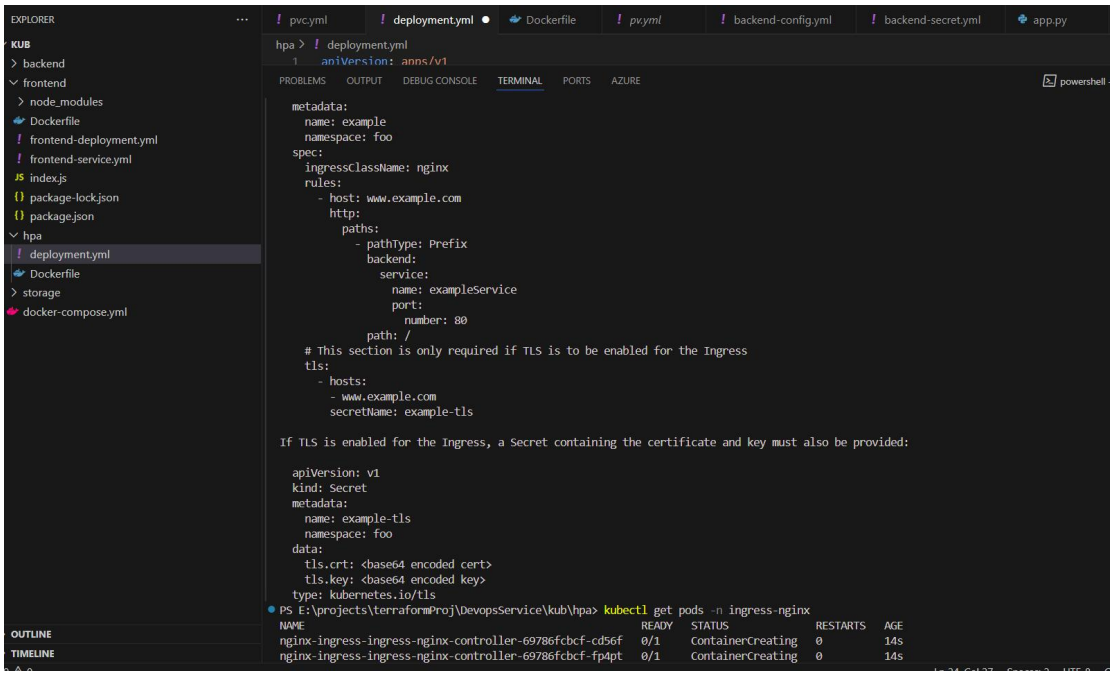
PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | AZURE

powershell - storage

```
backend-deployment-885cd577d-clvb5      0/1     ImagePullBackOff   0               80d
frontend-deployment-5c54749bf-zpmxc     0/1     ImagePullBackOff   0               80d
hello-world-bf87db69c-xq572             1/1     Running            1 (4h36m ago)   5h5m
my-nginx-deployment-7b7f68d87b-2dq8g    0/1     ImagePullBackOff   0               80d
my-nginx-deployment-7b7f68d87b-96p8q    0/1     ImagePullBackOff   0               80d
worker-67547f5d7b-zd6w8                 0/1     ImagePullBackOff   0               76d
PS E:\projects\terraformProj\DevopsService\kub\storage> kubectl exec -it backend-5d5844658c-v6kst -- ls -lah /var/log/app
total 8.0K
drwxrwxrwx 2 root root 4.0K Mar  6 14:56 .
drwxr-xr-x 1 root root 4.0K Mar  6 15:03 ..
PS E:\projects\terraformProj\DevopsService\kub\storage> kubectl exec -it backend-5d5844658c-v6kst -- sh -c 'echo "Test log entry" >> /var/log/app/test.log'
Test
PS E:\projects\terraformProj\DevopsService\kub\storage> kubectl exec -it backend-5d5844658c-v6kst -- ls -lah /var/log/app
total 8.0K
drwxrwxrwx 2 root root 4.0K Mar  6 14:56
```

Practical Task 4: Scale Applications with Horizontal Pod Autoscaler





Practical Task 5: Implement Ingress for External Access

```
3   metadata:

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   AZURE                                   powershell - ingress

Error: INSTALLATION FAILED: repo ingress-nginx not found
PS E:\projects\terraformProj\DevopsService\kub\hpa> helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories
PS E:\projects\terraformProj\DevopsService\kub\hpa> helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "secrets-store-csi-driver" chart repository
...Successfully got an update from the "csi-secrets-store-provider-azure" chart repository
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "argo" chart repository
...Successfully got an update from the "prometheus-community" chart repository
...Successfully got an update from the "grafana" chart repository
Update Complete. ⎈Happy Helming!⎈
PS E:\projects\terraformProj\DevopsService\kub\hpa> helm install nginx-ingress ingress-nginx/ingress-nginx --set controller.replicaCount=2 --namespace ingress
-nginx --create-namespace
NAME: nginx-ingress
LAST DEPLOYED: Thu Mar  6 17:55:03 2025
NAMESPACE: ingress-nginx
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The ingress-nginx controller has been installed.
It may take a few minutes for the load balancer IP to be available.
You can watch the status by running 'kubectl get service --namespace ingress-nginx nginx-ingress-ingress-nginx-controller --output wide --watch'

An example Ingress that makes use of the controller:
  apiVersion: networking.k8s.io/v1
  kind: Ingress
  metadata:
    name: example
    namespace: foo
  spec:
    ingressClassName: nginx
    rules:
```

File Edit Selection View Go Run Terminal Help kub

EXPLORER

- KUB
  - backend
    - app.py
    - backend-configmap.yml
    - backend-deployment.yml
    - backend-secret.yml
    - backend-service.yml
    - Dockerfile
    - requirements.txt
  - frontend
    - node_modules
    - Dockerfile
    - frontend-deployment.yml
    - frontend-service.yml
    - index.js
    - package-lock.json
    - package.json
  - hpa
    - deployment.yml
    - Dockerfile
  - ingress
    - ingress-class.yaml
    - ingress.yml
  - storage
  - docker-compose.yml

pvc.yml | deployment.yml | ingress.yml | ingress-class.yaml | Dockerfile | pv.yml

ingress > ingress.yml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: 192.168.49.2.nip.io
    http:
      paths:
      - path: /frontend
        pathType: Prefix
        backend:
          service:
            name: frontend-service
            port:
              number: 80
      - path: /backend
        pathType: Prefix
        backend:
          service:
            name: backend-service
            port:
              number: 80
```

```yaml
metadata:
  name: app-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
    tls.key: <base64 encoded key>
  type: kubernetes.io/tls
    tls.key: <base64 encoded key>
    tls.key: <base64 encoded key>
  type: kubernetes.io/tls
    tls.key: <base64 encoded key>
  type: kubernetes.io/tls
PS E:\projects\terraformProj\DevopsService\kub\hpa> kubectl get pods -n ingress-nginx
    tls.key: <base64 encoded key>
  type: kubernetes.io/tls
    tls.key: <base64 encoded key>
    tls.key: <base64 encoded key>
    tls.key: <base64 encoded key>
    tls.key: <base64 encoded key>
  type: kubernetes.io/tls
PS E:\projects\terraformProj\DevopsService\kub\hpa> kubectl get pods -n ingress-nginx
NAME                                                  READY   STATUS              RESTARTS   AGE
nginx-ingress-ingress-nginx-controller-69786fcbcf-cd56f   0/1     ContainerCreating   0          14s
nginx-ingress-ingress-nginx-controller-69786fcbcf-fp4pt   0/1     ContainerCreating   0          14s
PS E:\projects\terraformProj\DevopsService\kub\hpa> kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
api-gateway-service   NodePort    10.110.241.61   <none>        8080:30001/TCP   76d
backend-service       ClusterIP   10.104.40.174   <none>        5000/TCP         80d
frontend-service      NodePort    10.99.106.47    <none>        80:31340/TCP     80d
hello-world-service   NodePort    10.104.184.23   <none>        80:30010/TCP     5h46m
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          83d
my-nginx-service      NodePort    10.98.223.222   <none>        80:30007/TCP     80d
worker-service        ClusterIP   10.102.76.12    <none>        8081/TCP         76d
```